CS 330 – Spring 2023 – Homework 09

Due: Tuesday 5/2 at 11:59 pm on Gradescope

Collaboration policy Collaboration on homework problems is permitted, you are allowed to discuss each problem with at most 3 other students currently enrolled in the class. Before working with others on a problem, you should think about it yourself for at least 45 minutes. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is strictly forbidden.

You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem. You must also identify your collaborators. If you did not work with anyone, you should write "Collaborators: none." It is a violation of this policy to submit a problem solution that you cannot orally explain to an instructor or TA.

Typesetting Solutions should be typed and submitted as a PDF file on Gradescope. You may use any program you like to type your solutions. LaTeX, or "Latex", is commonly used for technical writing (`overleaf.com` is a free web-based platform for writing in Latex) since it handles math very well. Word, Google Docs, Markdown or other software are also fine.

Solution guidelines For dynamic programming algorithms, follow the solution guideline laid out in class:

1. precisely define the sub-problem with proper indexing
2. give the recursive formula and argue about its correctness
3. write the DP algorithm
4. write an algorithm that prints the elements in the optimal solution

You may use anything we learned in class without further explanation. This includes using algorithms from class as subroutines, stating facts that we proved in class, e.g. correctness of subroutines, running time of subroutines and use the notation. Your description should be at the level that it is clear to a classmate who is taking the course with you.

You should be as clear and concise as possible in your write-up of solutions.

A simple, direct analysis is worth more points than a convoluted one, both because it is simpler and less prone to error and because it is easier to read and understand.

(*) It is fine if the English description concentrates on the high level ideas and doesn't include all the details. But the reader should not have to figure out your solution solely based on the pseudocode. You can also add comments to your pseudocode, in fact that is best practice.

Problem 1. Tight edges

We are given a flow network $G(V, E, s, t, c)$ and the maximum flow f. (Note that f is the actual flow, not just its value.)

1. (Not to be handed in) How can you verify in time $O(n + m)$ that f is indeed a maximum flow?
2. an edge is called *tight* if decreasing the capacity of this edge is decreasing the value of the maximum flow.

   Describe an algorithm to identify one such tight edge.
3. Now we have the same input as before and in addition the edge (u, v) (given by its node ids). Suppose that the capacity of this edge is *decreased* by one, thus the new capacity is $c'(u, v) = c(u, v) - 1$.

   Describe what the new value of the maximum flow can be after the decrease. Give a one sentence explanation when each of the values occur.
4. Suppose that the value of the flow does change as the result of the capacity decrease. Design an O(n+m) algorithm to update the maximum flow.

Problem 2. Trains.

You are the police chief in a far away country. Recently a very valuable painting has been stolen from the city of Southton. You learn that the thieves will try to smuggle it to Trent City by train. You intend to send police officers to train stations to search all trains passing through them. Unfortunately your department has limited resources and you don't have enough officers to cover each station. So you decide that instead of searching every train you make sure that every possible train route is covered at least once.

As input you are given the rails map showing stations and which stations are directly connected by rails. You may assume that rails are one-directional. (If there is traffic in both directions that would be represented by two rails in opposite directions between the stations.) For this problem you don't need to know how much traffic each stretch of rail can carry*.

Design an efficient algorithm to find a minimum set of stations to send police officers so that every train route is intercepted.

**If this makes it easier for you, you may assume that the limit on each rail stretch is (number of stations) +1, it doesn't change the solution.*

1. (not to be handed in) Decide how to represent the rail network in general as a network flow problem.
2. Clearly describe the network flow graph that you use to model the problem of finding the minimum set of stations. (This is most likely a slight modification of the model you found in the previous point.)

3. What is the *exact* number of nodes and edges in your graph as a function of number of stations and rail stretches in the input?

4. Describe the algorithm to identify the stations.

5. What is the running time as a function of number of stations and rail stretches in the input?

6. Prove that your algorithm indeed finds a minimum set of stations.

Problem 3. Design your own graph.

As input you are given the integers n and d.

1. Design a network flow graph with $n = 8$ nodes (sourse s, sink t and $n - 2$ intermediate nodes) and such that each node has outdegree $d = 3$ and each edge capacity is 1. Specify the nodes, edges and edge capacities, such that the max flow in this network is[1]

   (a) as small as possible.

   (b) as large as possible.

   You don't have to prove that the flow is indeed the largest or smallest possible, but points will be deducted if your example is not min/max.

   (c) State the maximum flow value of both and identify a min cut.

2. What is the total number of nodes an edges of such a graph for general n and d? State the running time of Ford-Fulkerson in such a graph if edge capacities are 1.

[1] We have included some example tex syntax at the end of the tex file if you want to use it for your graph drawing. Including a hand drawn pdf is also fine.